Intelligent Robotic Navigation

Siddhartha Tripathy

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Intelligent robotic navigation

by,

Siddhartha Tripathy(Roll no:111CS0056)

of

Computer Science and Engineering Department

under the supervision of

Dr. Pankaj K. Sa

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

March' 2015

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India.

Certificate

This is to certify that this is a bonafide record of the project presented by *Siddhartha Tripathy ,111CS0056* (2014-2015) in partial fulfilment of the requirements of the degree of Bachelor of Technology in Computer Science and Engineering.

Date:

Dr. Pankaj K. Sa
(Project Guide)

Acknowledgment

First of all, I would like to express my deep sense of respect and gratitude towards my advisor and guide Prof Pankaj K. Sa, who has been the guiding force behind this work. I am greatly indebted to him for his constant encouragement, invaluable advice and for propelling me further in every aspect of my academic life. His presence and optimism have provided an invaluable influence on my career and outlook for the future. I consider it my good fortune to have got an opportunity to work with such a wonderful person. I would like to thank all faculty members and staff of the Department of Computer Science and Engineering, N.I.T. Rourkela for their generous help in various ways for the completion of this thesis. I would like to thank my friend Sampad Bhusan Mohanty, of Department of Electrical Engineering for helping me in every possible way and for the thoughtful and mind stimulating discussions we had, which prompted me to think beyond the obvious.

Abstract

Computer vision is a field of computer science which bridges the gap between hardware and software and the first step towards automation. It deal with the area of machines dependent on motions and their motion strictly rely on the vision i.e. camera. It is also the process of regenerating the physical scenario in computer. It also helps in understanding physical scenario and objects. The scientific aspect of the computer vision deals with extracting information as easily as possible even in conducive environment. In automated system, Computer vision plays a pivotal role while making decisions for the system. It helps in creating map of the environment the system is in. An effective vision system is capable in identifying its environment and its objects effectively and to pass the factors affecting the decisions to the system which will decide the course of action. With limited field of vision it becomes a challenging task for vision part to take into account of all the objects in the area. This is where artificial intelligence comes into play, which predicts expected action when system is provided with similar situation to deal with. By the help of computer vision localisation and mapping can be easily done and to act accordingly. The main objective of the project to develop a self-guided bot which is accompanied by its vision will be able to traverse from a given starting point to the destination point by tackling obstacles. It will be guided throughout the process by GPS coordinates from cellular phone to keep it on the track and vision to deal with static as well as dynamic obstacles.

Contents

List of Figures

Chapter 1

Introduction

Intelligent navigation for robot has been developed on the basis of the application of fuzzy logic, as fuzzy logic plays a crucial part in control of bots. Whenever there occurs a conflict of decision fuzzy system determines an intermediate solution for it. The navigation system helps in determining a path from source to destination taking into account of static as well as moving obstacles. While travelling a predefined path with expected obstacles is simple, when the bot is subjected to an unknown environment bot has to adapt to the environment for which learning technique can also be used. With the assistance of GPS and vision system, the whole environment can be plotted in 3 dimensional space which will make the decision making easier. Robotic navigation can be categorized into 2 categories for simplicity purpose, namely global based model and local based model. While the former one is to plan the path from source to destination with help of sequence of waypoints and the later one deals with immediate decisions which are to be made during obstacle avoidance.

1.1 Motivation And Objective

Intelligent robotic navigation is the need of the hour for humanity. Be it extra-terrestrial navigation or deep water navigation, wherever it is humanly impossible to operate,intelligent navigation of robot can be done.

- Developing an autonomous car which can traverse from one point another point specified by the GPS beforehand.
- Designing the required software that will guide the bot from source to destination.
- To tackle the obstacles computer vision has been used through Kinect, ultrasonic sensor for immediate obstacle detection.
- Using image processing determine the actual path and edge of the road so that it wont go out of track.

Chapter 2

Basic Concept

2.1 Computer Vision

Vision is a field that incorporates routines for procuring, transforming, investigating, and comprehending pictures and, when all is done, high-dimensional information from this present reality so as to create numerical data, e.g., in the manifestations of choices. Computer Vision Takes the 3 dimensional imagery and converts it 2 dimension. For motion based system, all valuable information which can be used for navigation is taken by computer vision.

Hence computer vision can also be described as the analysis based study of information acquired from different sources and by performing different algorithmic operations, it tries to acquire meaningful information which can be and will be used for bot's decision making.

Applying models and concepts related to artificial intelligent system computer vision generates computer vision based systems. In the context of this project, computer vision is used for localising and mapping the bot simultaneously and avoiding obstacles.

In robotics computer vision has to process information and produce

desired information which will help the system to make decision, as soon as it gets. In real time robust algorithms with high computation can't be implemented, as for processing each frame we will have very limited amount of time.

2.2 openCV

Opencv has been the foremost open source platform for image processing. It can perform low level operations such binarisation, thresholding and different types of thresholding on images. With increasing popularity it now supports high level image operations such as kalman filtering, machine learning, optical flow calculations, video calibration, video tracking etc. With support for C++ it is faster from other image processing tools and with support for currently popular python programming language, it makes image processing easy to implement.

Chapter 3

Literature Survey

Literature The very first modern UGV with was developed in 1960s by DARPA's A.I wing, namely **shakey** which was capable of processing English commands but was a touted as a failure during its time.

Under the leadership of Hans Moravec, stanford developed **stanford cart** which was based on obstacle avoidance technique using stereoscopic vision of a camera. This efficient for obstacle detection and avoidance, but lacked the speed. Other such UGV which was constructed a purpose of research was **HILARE** of France.

DARPA again came into existence in the context of intelligent robotic navigation during early 1980s with its new project autonomous land vehicle. It was fast (as speed was from 18 mph to 45 mph), capable of carrying weights sizeable amount.

In 2004, DARPA organised intelligent ground vehicle competition (IGVC), with the expectation of huge participation from top notch universities in developing an unmanned off road vehicle which can travel a distance 20km. Stanford university **stanley** won the competition by defeating Gerogia Tech university. [1][2]

Chapter 4

Hardware and Software requirement

4.1 Hardware requirement

4.1.1 Microsoft Xbox One kinect sensor

Microssoft Xbox One Kinect sensor is the 2nd phase sensor for Xboxs gaming console.The first sensor of Xbox 360 garnered popularity for its excellent depth and colour sensor. In xbox One Kinect sensor the limitations and glitches of xbox 360 kinect has been removed. It has got 2 stream namely colorstream and depthstream. The normal camera acts as a webcam and the IR sensor enabled depth sensor can be used for depth measurement. The frame per second of capturing the depth as well the depth is 30. The Infrared emitter keeps on sending IR waves of certain pattern and from it the depth of environment is determined. It has been calculated that IR sensor sends 50000 dots and calculates the depth. New Kinect sensor also done away with the motion blur with successfully reducing it to 14ms from 65 ms.[3]Kinect function flow chart 4.1

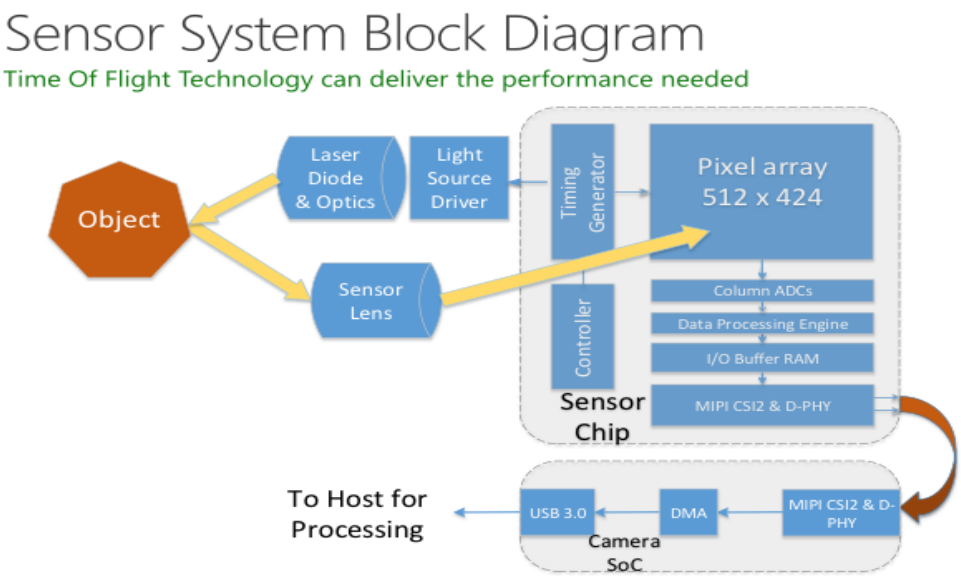

Figure 4.1: Kinect sensor functionality

4.1.2 Arduino Mega 2560 Development Board

It is used to receive driving signals from computer. It controls servo Motor and the relays connected to the vehicle circuit. It is also used to transfer different sensor data from phone to laptop for processing.Specifications includes 256 KBs of space with 8 KB with input voltage of 7-12V. Arduino mega 2560 4.2

4.1.3 Arduino Nano

It will be used for sending data from two ultra sonic sensor mounted in front of the bot through the bluetooth module to the laptop. The specification of arduino nano includes 16 KB or 32 KB of memory based on, if it is modelled upon atmega 168 or atmega 328 with input voltage of 7-12V.Arduino Nano 4.3

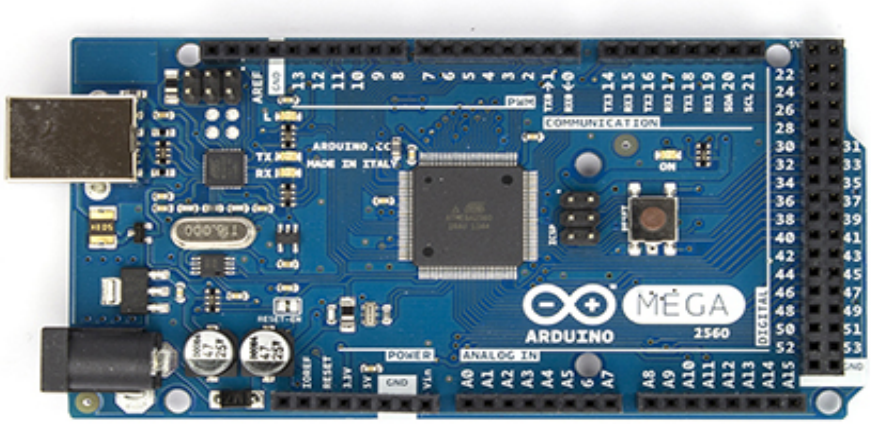

Figure 4.2: Arduino Mega 2560 Board

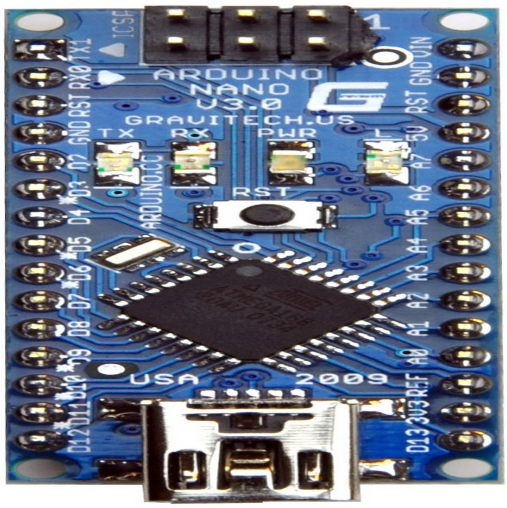

Figure 4.3: Arduino Nano Board

4.1.4 Ultra Sonic Sensor

Ultrasonic distance sensor provides stable and accurate distance measurements from 2cm to 450cm. It has a focus of less than 15 degrees and an accuracy of 2mm.Two of it will be mounted on front side of the bot which will detect obstacle very close to the bot and cant

be detetcted by ultra sonic sensor.Ultra sonic distance sensor 4.4

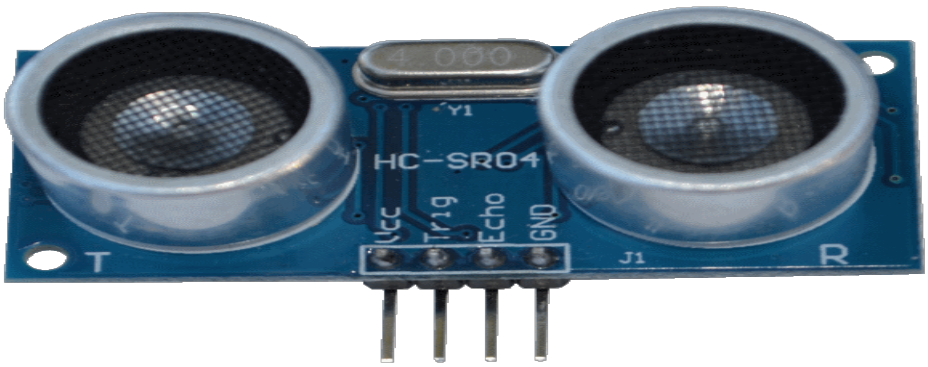

Figure 4.4: Ultra Sonic Sensor

4.1.5 steering servo

Steering servo comes with a inbuilt motor which with the help of gears rotates according to a given angle. Servo is fitted with a rotating shaft which,when tied with bots front wheel can rotate it any specific angle.Steering servo 4.5

4.2 Software Requirement

- Windows Operating System
- OpenCV Library

Figure 4.5: steering servo

- Kinect SDK for Kinect
- Visual studio
- Arduino IDE

Chapter 5

Software Implementation

The path planning for the Bot can be divided into two parts:

- Lane detection
- obstacle detection

For Lane detection:

- stablize the video using kalman filter
- generate the bird's eye view
- find the lane using hough transform

For obstacle detection:

- remove the noise from the depth map and apply morphologic operation on it.
- detect the obstacle if it is in the range.
- Perform appropriate Decision.

5.1 Integrating opencv Kinect SDK in Visual studio

OpenCV 2.0 is integrated with Visual Studio 2012. After the installation of Visual Studio is complete, for an empty project the debugger is set to win64 and in the VC++ directories include folder of both Kinect and opencv libraries are added. Then to the linker settings, the library files are added, so that the processes are linked to these libraries while execution. Linker Settings are changed by adding the all the following directories one by one. All the libraries files are added as shown below:

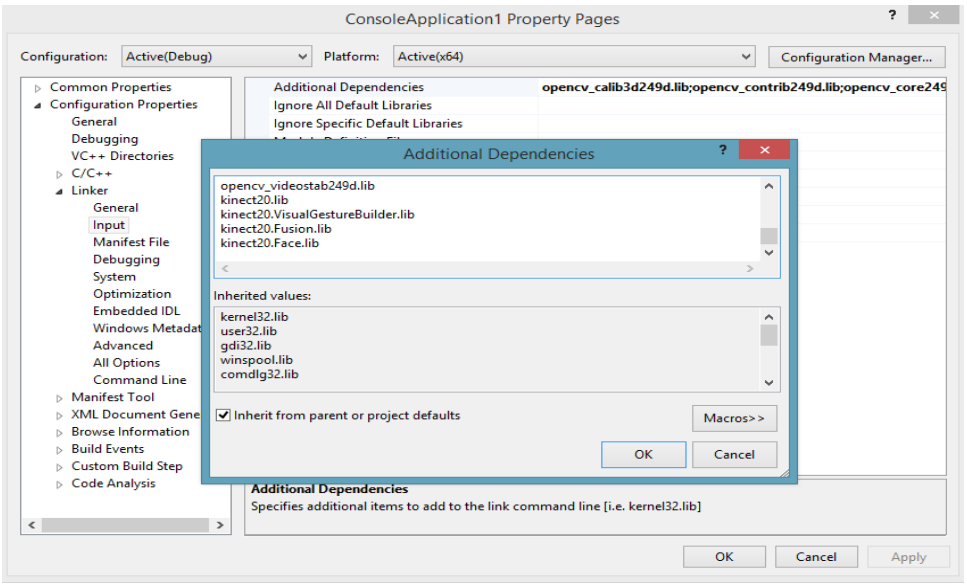

Figure 5.1: Visual studio setup

5.2 KML waypoint tracking

Keyhole markup language(KML) is an open source XML based system which is used for geographical location marking both for 2 dimensional as well as three dimensional map. It has been used by Goole since its acquisation in 2004, for google earth. A standard KML is parsed to generate waypoints of from the source to destination.After parsing the KML file the GPS waypoints are generated whicha re stored in a text file for kalman filter operation. A standard KML file is shown in 5.2 :

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<Placemark>
  <name>New York City</name>
  <description>New York City</description>
  <Point>
    <coordinates>-74.006393,40.714172,0</coordinates>
  </Point>
</Placemark>
</Document>
</kml>
```

Figure 5.2: KML file format

5.3 Bearing Calculation

It is the angle of rotation needed from one waypoint to another point in a given source to destination path. Using the following formula it is being calculated:

$$\theta = \arctan\left(y_1 - y_1, \cos\left(y_2 + y_1\right)/2\right)$$

Angle w.r.t true north is

$$\alpha = (90 - \theta)\%360$$

5.4 Hacking the Kinect Xbox One sensor

Kinect sensor uses both colorstream and depthstream for getting color image and depth image respectively. So Kinect can generate Color image(5.3), depth image(5.4), infrared image(5.5)

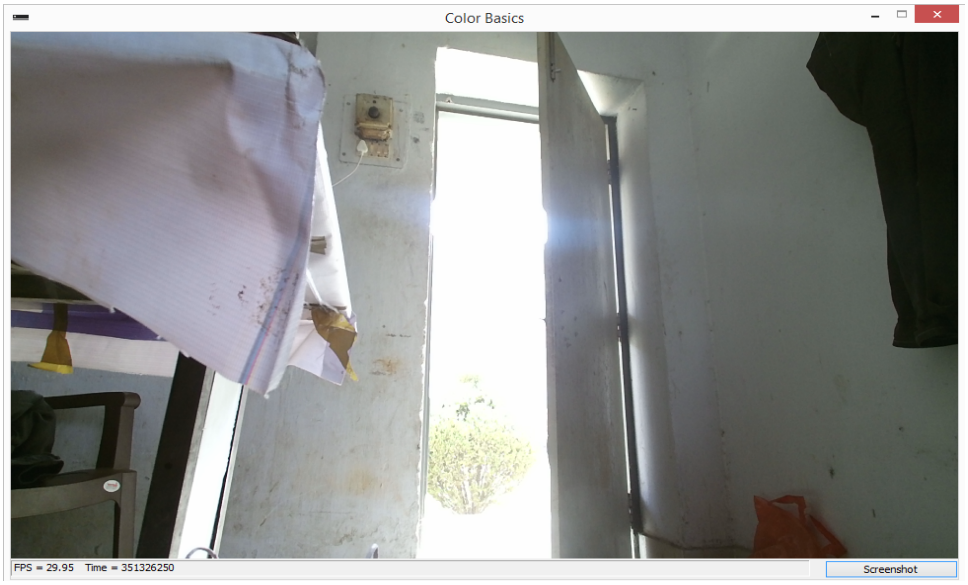

Figure 5.3: Color Image

5.5 Lane detection

For lane detection the color image is used, which is first converted to a gray scale image using canny operator and then houghline transform with probability is applied. The acquired points are then plotted over the output image stream. as shown in fig 5.6

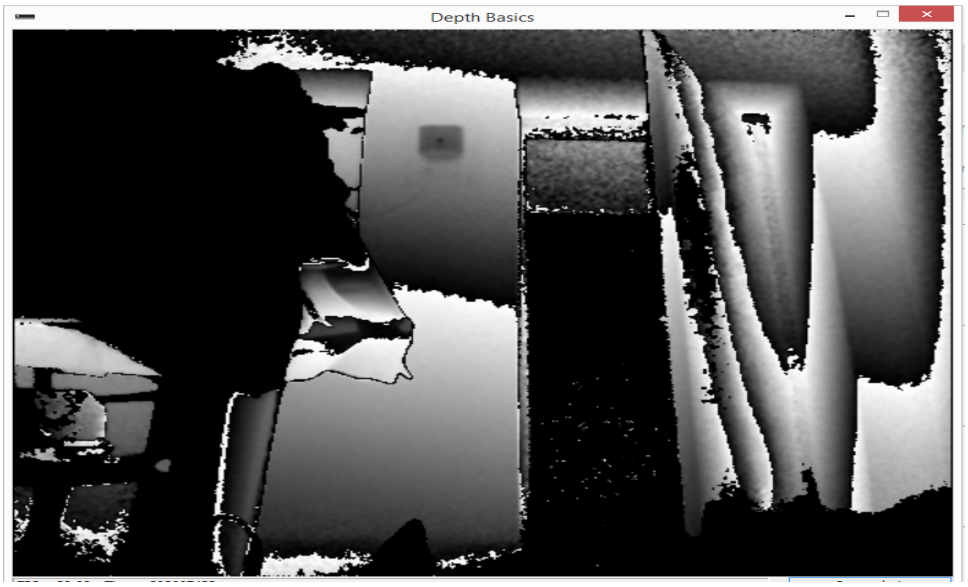

Figure 5.4: Depth Image

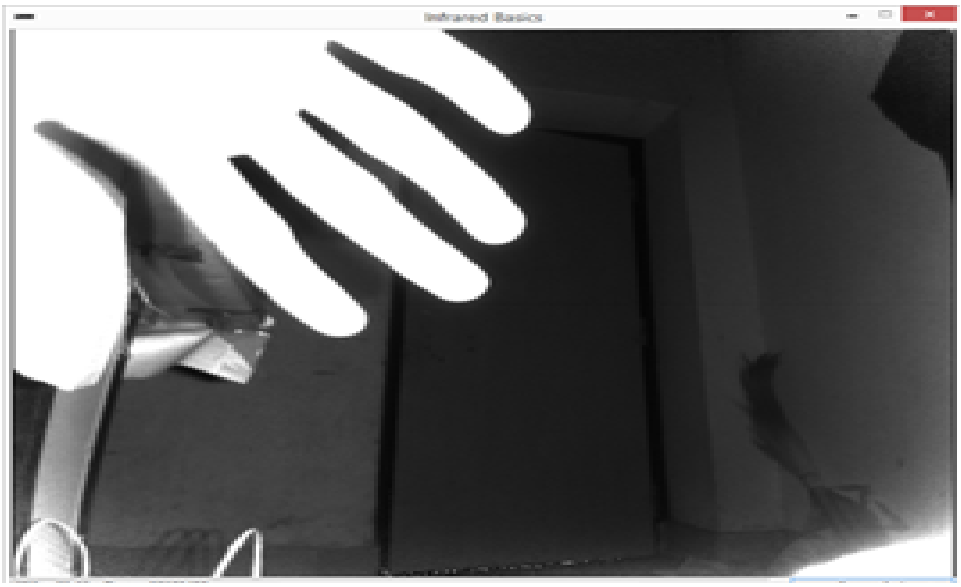

Figure 5.5: Infrared Image

5.6 Kalman Filter

Kalman filter is an algorithm which takes input from the current state and predicts the expected next state when it is provided with a possible error. When the error is of the form of Gaussian distribution kalman filter works the best as it minimizes the mean square error in the

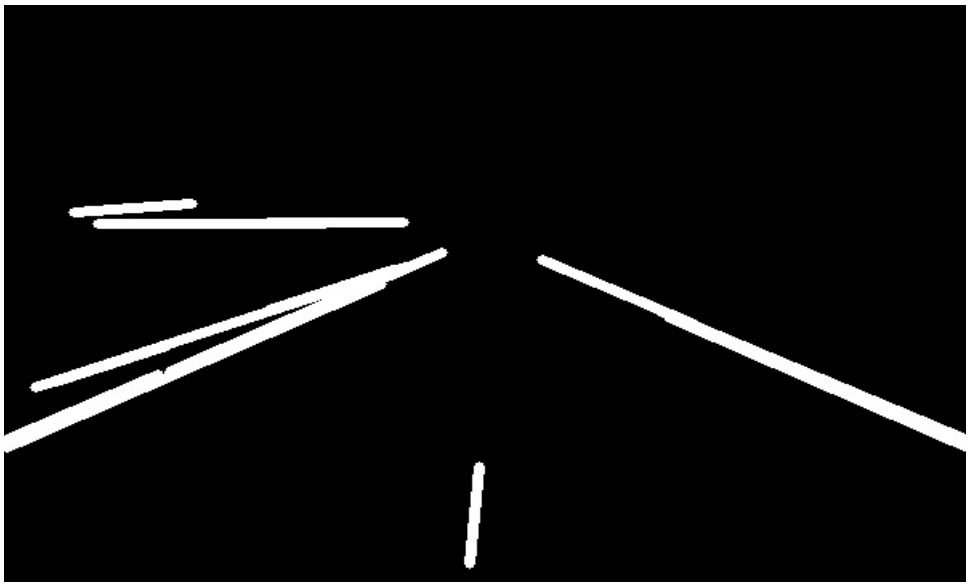

Figure 5.6: Lane detection

system. It is a linear model which is associated with mean and standard deviation. Kalman filter works in two steps. The first step being the prediction step where it generates the possible current state values with the expected error. When the output of the next is provided, kalman filter updates its covariance matrix accordingly. Advantages of Kalman filter over any other filter like particle filter is, it can be used in real time processing. It doesnt require extra information to predict the future states. For the purpose of GPS tracking and video stabilization a linear kalman filter is used. As the noise distribution is tested to be Gaussian type. While particle filter works better when the noise is not Gaussian and it is more robust. A standard Kalman filter equation is:

$$X_k = F_k + X_{k-1} + B_k + u_k$$

Where Fk is the state transition model which is applied to the previous statexk1; Bk is the control-input model which is applied to the control

vector u_k; W_k is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_k.

$$y_k = z_k - H_k X_{k|k-1}$$

.

5.6.1 Kalman filter for GPS Tracking

For GPS tracking, kalman filter can be efficiently used to predict the next state as the process noise is Gaussian in nature. When the input is provided as latitude and longitude with a given accuracy to start with kalman filter measures the next possible position of the state and calculates the variance.[4] as shown in the code snippet 5.7

```
KalmanLatLong(float Qmps)
{  this->Qmps = Qmps;
   variance = -1;}
long get_TimeStamp()
{  return timems;}
double get_lat()
{  return lat;}
double get_lng()
{   return lng;}
float get_accuracy()
{   return (float)sqrt(variance);}
void SetState(double lat, double lng, float accuracy, long timems)
{  this->lat=lat;
   this->lng=lng;
   variance=accuracy*accuracy;
   this->timems=timems;}
void Process(double latmsmnt,double lngmsmnt,float accuracy,long timems)
{
    if (accuracy<MinAccuracy)
        accuracy=MinAccuracy;
    if (variance < 0)
    {   this->timems=timems;
        lat=latmsmnt;
        lng=lngmsmnt;
        variance=accuracy*accuracy;}
    else
    {   long timems=timems-this->timems;
    if (TimeInc_milliseconds>0)
    {   variance+=timems*Qmps*Qmps/1000;
        this->timems=timems;
        }
        float K=variance/(variance+accuracy*accuracy);
        lat+=K*(latmsmnt-lat);
        lng+=K*(lngmsmnt-lng);
        variance=(1-K)*variance;
```

Figure 5.7: Kalman filter for GPS tracking

5.6.2 Kalman Filter Path stabilization

Kalman filter can be used to stabilize a video. For video stabilization the following course of operations can be done Each time while

processing an input is set for current frame and another one for the previous frame. There after all the good features i.e. all the positions that change during the frame transition is reported. The optical flow of the video is then measured to find the possible movement of all feature points. The points having only considerable amount of changes is stored, i.e. only good feature points are stored. Now to find the movement along x and y axis and the rotation in the input, rigid transformation is done, which measures dx(change in x),dy(change in y) and da(change in angle). This rigid transformation function nullifies all the movements to adjust the video. There after the output needs to be smoothed so that video wont decrease in size further. For this kalman filter is used. It takes the changes in the position and angle and smoothes the output to produce smoothed output. As it can be seen from the video stabilization adds a thin black line to compensate for the movements occur in the original video.

The kalman estimated output is then wrapped with the original video.[5] For this wrapAffine function is used. as the output is shown in 5.8

5.7 Birds eye view generate

Birds eye view is essential for robotic navigation purposes. To find the actual distance between obstacle and the bot, it needs to be the birds eye view to do the job.[5]as shown in code snippet 5.9 and the output in fig 5.10

Figure 5.8: Road stablisation

```
// from top-left in clockwise order

outputQuad[0] = Point2f(152,78);
outputQuad[1] = Point2f(433,68);
outputQuad[2] = Point2f(628, 205);
outputQuad[3] = Point2f(20,221);
//  from top-left in clockwise order
inputQuad[0] = Point2f( 3*input.cols/8,3*input.rows/8 );
    inputQuad[1] = Point2f( 5*input.cols/8,3*input.rows/8);
    inputQuad[2] = Point2f( 5*input.cols/8,5*input.rows/8);
    inputQuad[3] = Point2f( 3*input.cols/8,5*input.rows/8 );
circle(input,outputQuad[0],10,(255,255,255));
circle(input,outputQuad[1],10,(0,0,0));
circle(input,outputQuad[2],20,(255,0,255));
circle(input,outputQuad[3],30,(255,100,100));

lambda = getPerspectiveTransform( inputQuad,outputQuad);

warpPerspective(input,output,lambda,Size(input.cols,input.rows),CV_INTER_LINEAR | CV_WARP_INVERSE_MAP );
```

Figure 5.9: Topview code

5.8 obstacle detection and avoidance

or each frame of depth, each pixel represents the corresponding distance from the Kinect, this data is accessed and stored for further processing such as static and dynamic obstacle detection.

The kinect sensor has a visiblity range from 45cm to 8 meter. so

Figure 5.10: topview

anything that comes between kinect and 45cm barrier it will show dark. This can be exploited to detect obstacle and to avoid it.

so the video stream is divided into 3 screen namely center,left and right. Whenever there comes an obstacle in center frame, the left and right screen will be checked and whichever is not showing any obstacle, the bot will drift that way. If both the left and right screen are also occupied by obstacles. The distance between center frame obstacle and both side screen obstacle will be calculated. Which ever distance is higher, bot will be drift that way.[6]

Figure 5.11: obstacle detection

Chapter 6

Hardware implementation

6.1 Connecting Microcontroller

Arduino Mega 2560 microcontroller connects to the Ultra sonic sensor which will send data to the computer for further processing and Speed encoder is also connected to it.

Arduino Nano handles all the driving data. All the driving instructions are sent to this microcontroller. Servo motor and motor driver circuits have been connected to it.

Bluetooth module One piece connected to the computer for communicating between phone and computer. The other one is to connect the Arduino Nano with the computer which will transfer data related to driving between computer and Arduino Nano.

The bot will perform operations such as move left,right,stop,start when it receives instruction from computer. The ultra sonic sensor data will be sent to the computer through the on board micro controller. as shown in the code snippet the low level drive instructions: The complete flow chart can be shown as:

```
void carDrive(String cmd , int value)
{
  /////    FOR TURNING ////
  if(cmd == "turn")
 { if((value < carServoMAX) && (value > carServoMIN))
    carServo.write(value);}
  //// FOR LINEAR MOTION ////
  else
  {
    if(cmd == "forward")
    {
      digitalWrite(carMotorPLUS,HIGH);
      digitalWrite(carMotorMINUS,LOW);   }
    else if(cmd == "back")
    { digitalWrite(carMotorPLUS,LOW);
      digitalWrite(carMotorMINUS,HIGH);   }
    else if(cmd == "stop")
     {digitalWrite(carMotorPLUS,LOW);
      digitalWrite(carMotorMINUS,LOW);   }
    else if(cmd == "brake")
    { digitalWrite(carMotorPLUS,HIGH);
      digitalWrite(carMotorMINUS,HIGH);   }
    // Update Speed value
    analogWrite(carMotorPWM,value);

  }
```

Figure 6.1: Low level drive

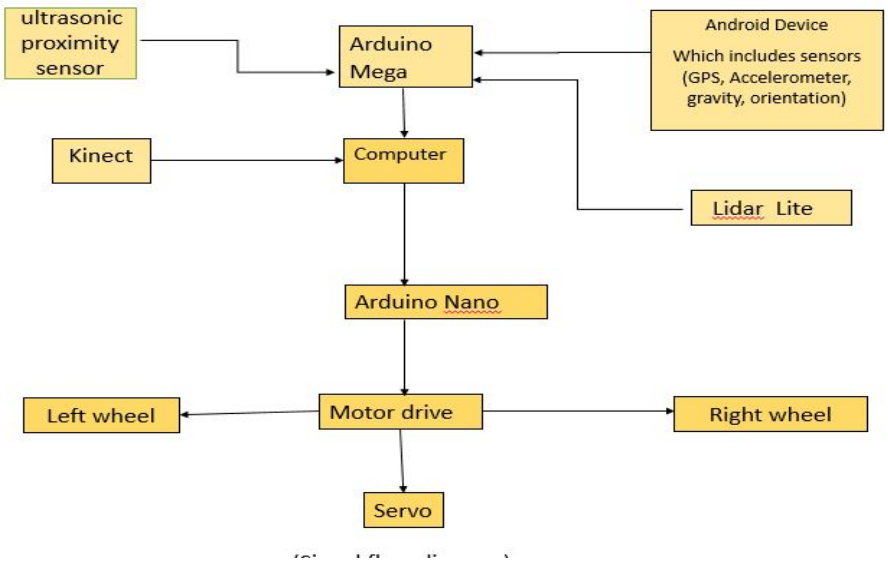

Figure 6.2: signal flow chart

Chapter 7

Conclusion and Future Work

Almost all part of the bot as described earlier has been integrated. The bot operates well with the help of ultra sonic sensor. With teh help of kinect the obstacle avoidance becomes more easy but sometimes the bot may fail to work due to absence of proper learning.

Proper simulation and learning is need to be done for the bot. Point cloud library can be used to map the entire space in 3D. The simulation will be done labview to test the corner cases where bot might fail to respond and proper learning is to be given. Kinects infrared can be exploited to work in domestic environment. For the bot to move faster during its traversal LiDAR(light RADAR) can be used,whose range is 40 meters to detect fast moving obstacles.

Bibliography

[1] Douglas W Gage. Ugv history 101: A brief history of unmanned ground vehicle (ugv) development efforts. Technical report, DTIC Document, 1995.

[2] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[3] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[4] Nopaddol Chadil, Apirak Russameesawang, and Phongsak Keeratiwintakorn. Real-time tracking management system using gps, gprs and google earth. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*, volume 1, pages 393–396. IEEE, 2008.

[5] Deepankar Purniya, Gigyanshu Pradhan, and Amit Nayak. *Robotic navigation in the presence of static and dynamic obstacles.* PhD thesis, 2012.

[6] Dmitri A Dolgov and Sebastian Thrun. Vehicle navigation system with obstacle avoidance, October 22 2007. US Patent App. 11/876,203.